

84
Q1
generating a signal using emulation hardware, wherein the signal indicates that the functional units are emulating an SSE instruction; and

sending the signal to the functional units, and
wherein the step of determining comprises determining after the signal is sent.

22. (New) The system of claim 12, wherein the processor is further configured to emulate an instruction set by:

generating a signal using emulation hardware, wherein the signal indicates that the functional units are emulating an SSE instruction; and

sending the signal to the functional units, and
wherein the step of determining comprises determining after the signal is sent.--

REMARKS

Claims 1, 3-15, and 17-22 are pending. By this amendment, claims 1, 10, and 12 are amended. Claim 16 is canceled. New claims 21-22 are added. No new matter is introduced. Reconsideration and allowance of the claims in view of the above-amendments and the remarks that follow are respectfully requested.

Claims 1, 3-8, 12-17, and 19 are rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. patent no. 6,330,657 B1 to Col, et al., (hereinafter "Col") in view of Hennessy and Patterson, Computer Architecture – A Quantitative Approach, 2nd Edition, 1996 (hereinafter "Hennessy"). Claim 9 is rejected under 35 U.S.C. § 103(a) as being unpatentable over Col in view of Hennessy and U.S. patent no. 6,038,652 to Phillips (hereinafter "Phillips"). Claim 18 is rejected under 35 U.S.C. § 103(a) as being unpatentable over Col in view of Hennessy and U.S. patent no. 6,321,327 to Makineni (hereinafter "Makineni"). Claims 10-11 remain rejected under 35 U.S.C. § 103(a) as being unpatentable over Abdallah1 in view of Abdallah2, cited in the First Office Action, paper no. 3. Claim 20 is rejected under 35 U.S.C. § 103(a) as being unpatentable over Abdallah1 in view of Abdallah2. These rejections are respectfully traversed.

Claim 1

Claim 1 has been amended to recite that the method executes the microinstructions in functional units of a floating point unit and determines whether an exception occurs before setting any result registers with results of the executing. Further, if an exception occurs in any of the microinstructions, then the microinstructions are not only canceled, but also the result registers may not be written. The result registers for all of the microinstructions are only set after determining that there are no exceptions taken in any of the microinstructions. Support for this amendment is found, for example, at page 8, lines 11-14 of the specification.

As amended, claim 1 overcomes the rejections, because the cited references do not teach or suggest preventing the results of the executing from being written to the result registers if an exception is taken in any of the microinstructions. Col does not teach anything regarding exception handling. Hennessy specifically teaches away from the language recited in amended claim 1. Hennessy teaches the prior art methods of exception handling, described in the background of the present specification. Specifically, Hennessy recognizes that, "By the time the fault is seen, several other instructions will be in execution." Hennessy, p. 182. Hennessy suggests forcing a trap and disabling writes "for the faulting instruction and for all instructions that follow in the pipeline," but does not teach or suggest how to handle the writing of results from co-pending microinstructions executing simultaneous with the microinstruction taking the fault. Hennessy, p. 183. To the contrary, Hennessy notes that existing systems processing floating-point exceptions have a particular difficulty because the faulting instruction "writes its result before the exception can be handled." Id. Although it is desirable to maintain the state of the machine, Hennessy recognizes that this is difficult to do with floating point exceptions, "Because floating-point operations may run for many cycles." Id. Hennessy's solution to this problem is that "the hardware must be prepared to retrieve the source operands, even if the destination is identical to one of the source operands." Id. In other words, the Hennessy system still allows the writing of results of instructions executing in simultaneous with the instruction taking the exception (and perhaps even results of the faulting instruction itself), but provides a back-off mechanism to correct the result register. In contrast, amended claim 1 does not allow the setting of any result register until it is known that none of the co-pending microinstructions will cause a fault. As a result, no back-off mechanism is required.

Because the cited references do not teach or suggest preventing the writing of result registers until after determining that no exception is taken, claim 1 as amended should be allowed. Claims 3-9 and 19 depend from claim 1 and for these reasons and for the other limitations they recite should be allowed. Reconsideration is requested.

Claim 10

Claim 10 has been amended to recite that the operations are executed simultaneously, in lockstep, similar to the limitation recited in claim 1 as added by the first amendment to claim 1 to overcome the Abdallah1 and Abdallah2 references. The cited references do not teach or suggest executing the instructions in lockstep. Claim 10 has further been amended in a manner similar to the second amendments to claim 1, discussed herein. Support for this amendment is found, for example, at page 8, lines 11-14 of the specification. The Office

Action notes that claim 10, as filed, could potentially include a back-off mechanism, shadow register, or similar means of “undoing” results of a microinstruction that is co-pending with the microinstruction that causes the exception. As amended, claim 10 obviates the need for a back-off mechanism because the results of the co-pending microinstruction are not committed to any result register until after it is determined that no exceptions exist. The cited references do not teach or suggest preventing the setting of all results registers until after determining that no exceptions are taken. For these reasons, amended claim 10 should be allowed. Claims 11 and 20 read on claim 10 and for these reasons and for the other limitations they recite should be allowed. Reconsideration is requested.

Claim 12

Claim 12 has been amended in a manner similar to twice-amended claim 1, and is allowable for the same reasons that claim 1 is now allowable. Support for this amendment is found, for example, at page 8, lines 11-14 of the specification. The cited references do not teach or suggest a floating point unit with a plurality of functional units that execute the microinstructions. Nor do the cited references teach or suggest preventing the setting of result registers for all of the functional units if any of the result registers takes an exception. To the contrary, the cited references teach against the language recited in claim 12. Claim 12 as twice amended is now allowable. Claims 13-18 and 22 depend from claim 12 and for these reasons and for the other limitations they recite should be allowed. Reconsideration is requested.

Claim 16

Claim 16 has been canceled for reasons unrelated to patentability.

New Claims

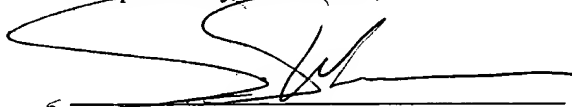
New claims 21 and 22 are added to more particularly claim the invention. Support for claims 21-22 is found, for example, in the specification at page 9, lines 14-25, and in Figure 4. The cited references do not teach or suggest generating a signal and sending the signal to the functional units to indicate that an SSE instruction is being emulated by the microinstructions executed in the floating point units. Nor do the cited references teach or suggest that the signal is used to trigger the step of determining whether an exception occurs and, if so, to inhibit the setting of result registers for all of the microinstructions. Claims 21-22 should be allowed.

CONCLUSION

In view of the above amendments and remarks, Applicant respectfully asserts that the application is in condition for allowance. Prompt reexamination and allowance of claims 1, 3-15, and 17-22 is respectfully requested.

Attached hereto is a marked-up version of the changes made to the claims by the current amendment. The attached pages are captioned "**Version with markings to show changes made.**" In addition, a clean copy of the pending claims is attached. The attached claims are captioned "**Pending Claims.**"

Respectfully submitted,



Date: **February 20, 2003**

Sean S. Wooden, Reg. No.: 43,997
DORSEY & WHITNEY LLP
1001 Pennsylvania Avenue, N.W.
Suite 400 South
Washington, DC 20004
Tel. (202) 442-3000
Fax (202) 442-3199

Attachment:

VERSION WITH MARKINGS TO SHOW CHANGES MADE

In the Claims:

Claim 16 has been canceled.

Claims 1, 10, and 12 have been amended as follows:

1. (Twice Amended) A method for processing software instructions comprising:[,]
 decomposing a macroinstruction into a plurality of microinstructions,
 issuing all of the plurality of microinstructions simultaneously, in parallel,
 executing all of the plurality of microinstructions simultaneously, in lockstep using functional units in a floating point unit,
 determining whether an exception occurs in any of the microinstructions, before writing results of the executing to result registers, and
 if an exception occurs in any of the microinstructions, canceling all of the microinstructions and preventing the results of the executing from being written to the result registers, and
 if no exception occurs in any of the microinstructions, writing the results of the executing to the result registers.
10. (Amended) A method for processing software instructions comprising:[,]
 [(a)] providing two microinstructions to emulate a high-half and a low-half SSE operation,
 [(b)] forcing the high-half and low-half operations to issue in parallel,
 [(c)] dispatching the high-half and low-half operations simultaneously to a first FP unit and to a second FP unit, respectively,
 executing the high-half and low-half operations simultaneously, in lockstep,
 [(d)] generating a signal from an emulator's hardware,
 [(e)] sending the signal to the first and second FP functional units,
 [(f)] determining whether an exception is taken in either the first or the second FP unit,
 [(g)] if an exception is taken in either the first or second FP unit, [flushing a result in the other FP unit, and]
 preventing results from the high-half and low-half operations from being written to result registers, and
 canceling both the high-half and low-half operations, and

[(h)] updating MXCSR flags based upon the results of the first and second FP units.

12. (Twice Amended) A computer system comprising:[,]

a processor comprising,

[(a)] a floating point unit comprising a plurality of functional units adapted to execute microinstructions;

[(b)] a ROM;

[(c)] a plurality of floating point registers;

wherein the processor is configured to emulate an instruction set by:

[(a)] decomposing a macroinstruction into a plurality of microinstructions;

[(b)] issuing all of the plurality of microinstructions simultaneously, in parallel, to the functional units,

[(c)] determining whether an exception occurs in any of the functional units,
[microinstructions, and]

setting result registers for results of each of the functional units only if no exception occurs in any of the functional units, and

[(d)] if an exception occurs in any of the microinstructions, canceling all of the microinstructions and preventing the setting of result registers for all of the functional units.

New claims 21-22 have been added as follows:

21. (New) The method of claim 1,

wherein the step of executing comprises executing using a plurality of functional units of a floating point unit,

further comprising:

generating a signal using emulation hardware, wherein the signal indicates that the functional units are emulating an SSE instruction; and

sending the signal to the functional units, and

wherein the step of determining comprises determining after the signal is sent.

22. (New) The system of claim 12, wherein the processor is further configured to emulate an instruction set by:

generating a signal using emulation hardware, wherein the signal indicates that the functional units are emulating an SSE instruction; and

sending the signal to the functional units, and

wherein the step of determining comprises determining after the signal is sent.